Class 13 RNA Seq Intro

Isabella Ruud PID: A59016138

Table of contents

In today's class we will analyze some published RNA seq experiments where airway smooth muscle cells were treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al., 2014)

We will use the DESeq2 package for the heavy lifting in a little bit, but first let's read the data and get to know how things work

Data import

There are two datasets needed for this type of analysis: -countData:the transcript abundances (read counts per gene) -colData: metadata about the columns in countData (ie experimental set up)

```
counts <- read.csv("airway_scaledcounts.csv", row.names=1)
metadata <-  read.csv("airway_metadata.csv")
```

```
head(counts)
```

```
                SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
ENSG00000000003        723        486        904        445       1170
ENSG00000000005          0          0          0          0          0
ENSG00000000419        467        523        616        371        582
ENSG00000000457        347        258        364        237        318
ENSG00000000460         96         81         73         66        118
ENSG00000000938          0          0          1          0          2
                SRR1039517 SRR1039520 SRR1039521
ENSG00000000003       1097        806        604
ENSG00000000005          0          0          0
ENSG00000000419        781        417        509
ENSG00000000457        447        330        324
ENSG00000000460         94        102         74
ENSG00000000938          0          0          0
```

```
head(metadata)
```

```
          id     dex celltype     geo_id
1 SRR1039508 control   N61311 GSM1275862
2 SRR1039509 treated   N61311 GSM1275863
3 SRR1039512 control  N052611 GSM1275866
4 SRR1039513 treated  N052611 GSM1275867
5 SRR1039516 control  N080611 GSM1275870
6 SRR1039517 treated  N080611 GSM1275871
```

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

There are 38694 genes in the dataset

Q2. How many control experiments do we have?

```
table(metadata$dex)
```

```
control treated
      4       4
```

```
sum(metadata$dex == "control")
```

```
[1] 4
```

There are 4 control experiments

Make sure that the columns of the count data are the same as the id column in the metadata

```
all(colnames(counts) == metadata$id)
```

```
[1] TRUE
```

Mean counts per condition

Let's find the average gene counts (ie rows) for control and treated conditions (ie columns)

Find the control samples and calculate the mean counts per gene across these samples

-extract all "control" columns/experiments -then find the row-wise average for these columns

> Q3.How would you make the above code in either approach more robust? Is there a function that could help here?

Can use rowMeans

```
control.inds <- metadata$dex == "control"
control.counts <- counts[,control.inds]
control.mean <- rowMeans(control.counts)
```

> Q4. Do the same for the treated columns to produce treated.mean

```
treated.inds <- metadata$dex == "treated"
treated.counts <- counts[,treated.inds]
treated.mean <- rowMeans(treated.counts)
```

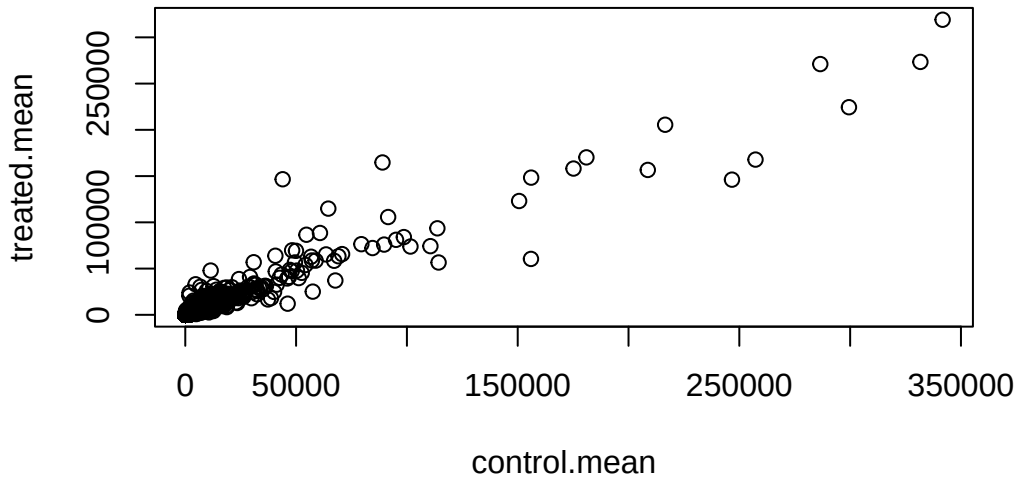
Let's store these mean values all in one dataframe

```
meancounts <- data.frame(control.mean, treated.mean)
head(meancounts)
```

```
                 control.mean treated.mean
ENSG00000000003       900.75       658.00
ENSG00000000005         0.00         0.00
ENSG00000000419       520.50       546.00
ENSG00000000457       339.75       316.50
ENSG00000000460        97.25        78.75
ENSG00000000938         0.75         0.00
```

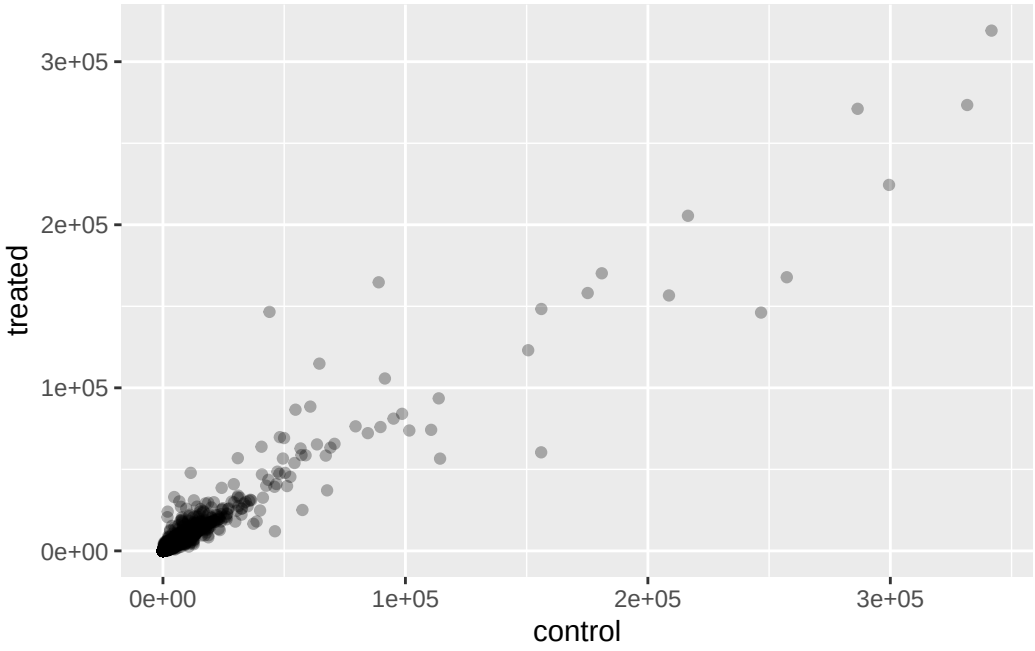
Q5 Create a scatter plot showing the mean of the treated samples against the mean of the control samples.

```
plot(meancounts)
```




```
library(ggplot2)

ggplot(meancounts) +
  aes(x = control.mean, y = treated.mean) +
  geom_point(alpha = 0.3) +
  labs(x = "control", y = "treated")
```



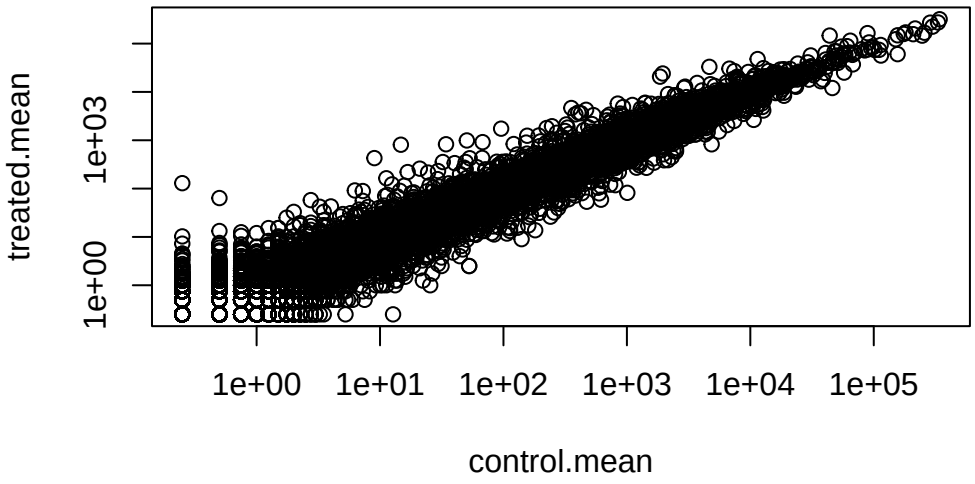
Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

log = "xy"

```
plot(meancounts, log = "xy")
```

```
Warning in xy.coords(x, y, xlabel, ylabel, log): 15032 x values <= 0 omitted
from logarithmic plot
```

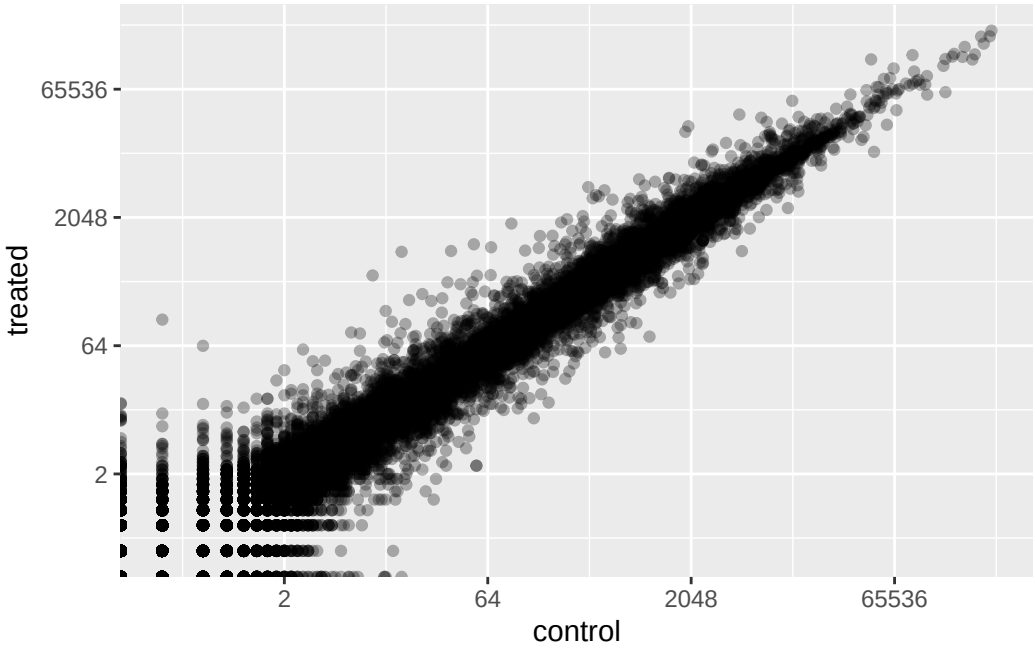
```
Warning in xy.coords(x, y, xlabel, ylabel, log): 15281 y values <= 0 omitted
from logarithmic plot
```




```
ggplot(meancounts) +
  aes(x = control.mean, y = treated.mean) +
  geom_point(alpha = 0.3) +
  labs(x = "control", y = "treated") +
  scale_x_continuous(trans="log2") +
  scale_y_continuous(trans="log2")
```

```
Warning in scale_x_continuous(trans = "log2"): log-2 transformation introduced
infinite values.
```

```
Warning in scale_y_continuous(trans = "log2"): log-2 transformation introduced
infinite values.
```



Log fold chage

We most often work in log2 units - why? Because the interpretation is much more straightforward.

```
log2(20/20)
```

```
[1] 0
```

```
log2(20/40)
```

```
[1] -1
```

```
log2(40/20)
```

```
[1] 1
```

Calculate log2 fold change (log2fc) of treated/control

```
meancounts$log2fc <- log2(meancounts$treated.mean / meancounts$control.mean)
head(meancounts)
```

```
                control.mean treated.mean      log2fc
ENSG00000000003       900.75       658.00 -0.45303916
ENSG00000000005         0.00         0.00         NaN
ENSG00000000419       520.50       546.00  0.06900279
ENSG00000000457       339.75       316.50 -0.10226805
ENSG00000000460        97.25        78.75 -0.30441833
ENSG00000000938         0.75         0.00        -Inf
```

There are some weird numbers in the log2fc values like -Inf and NaN all because there are zero count genes. I need to filter these out (ie remove them) before going any further

```
to.keep <- rowSums(meancounts[,1:2] == 0) == 0
mycounts <- meancounts[to.keep,]
```

How many non zero count genes do we have now?

```
nrow(mycounts)
```

```
[1] 21817
```

21817 genes are left

> Q7. What is the purpose of the arr.ind argument in the which() function call above? Why would we then take the first column of the output and need to call the unique() function?

The arr.ind argument makes it so that the places where there is a true value is returned as a matrix (ie both row and column position). The first column contains the genes to remove and unique() is called to make sure that if a gene has two 0s, it does not try to double-remove it

> Q8 How many genes are up regulated at a log2fc > 2?

```
sum(mycounts$log2fc>2)
```

```
[1] 250
```

```
paste(sum(mycounts$log2fc>2) / nrow(mycounts) * 100, "%")
```

```
[1] "1.14589540266764 %"
```

250 genes

Q9 How many genes are down regulated at a log2fc < -2?

```
sum(mycounts$log2fc < (-2))
```

```
[1] 367
```

```
paste(sum(mycounts$log2fc < (-2)) / nrow(mycounts) * 100, "%")
```

```
[1] "1.6821744511161 %"
```

367 genes

Q10 Do you trust these results?

No, because with over 21,000 genes, there are going to be genes that randomly have different log2 fold changes so you want to find the genes that are actually differentially regulated and have statistical significance

DESeq analysis

To do this analysis properly, we can use the BioConductor package DESeq2

```
#! message: false
library(DESeq2)
```

```
Loading required package: S4Vectors
```

```
Loading required package: stats4
```

```
Loading required package: BiocGenerics
```

```
Attaching package: 'BiocGenerics'

The following objects are masked from 'package:stats':

    IQR, mad, sd, var, xtabs

The following objects are masked from 'package:base':

    anyDuplicated, aperm, append, as.data.frame, basename, cbind,
    colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
    get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
    match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
    Position, rank, rbind, Reduce, rownames, sapply, saveRDS, setdiff,
    table, tapply, union, unique, unsplit, which.max, which.min


Attaching package: 'S4Vectors'

The following object is masked from 'package:utils':

    findMatches

The following objects are masked from 'package:base':

    expand.grid, I, unname

Loading required package: IRanges


Attaching package: 'IRanges'

The following object is masked from 'package:grDevices':

    windows

Loading required package: GenomicRanges

Loading required package: GenomeInfoDb
```

```
Loading required package: SummarizedExperiment

Loading required package: MatrixGenerics

Loading required package: matrixStats


Attaching package: 'MatrixGenerics'

The following objects are masked from 'package:matrixStats':

    colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
    colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
    colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
    colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
    colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
    colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
    colWeightedMeans, colWeightedMedians, colWeightedSds,
    colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
    rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
    rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
    rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
    rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
    rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
    rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
    rowWeightedSds, rowWeightedVars

Loading required package: Biobase

Welcome to Bioconductor

    Vignettes contain introductory material; view with
    'browseVignettes()'. To cite Bioconductor, see
    'citation("Biobase")', and for packages 'citation("pkgname")'.


Attaching package: 'Biobase'

The following object is masked from 'package:MatrixGenerics':

    rowMedians
```

```
The following objects are masked from 'package:matrixStats':

    anyMissing, rowMedians
```

Like most BioConductor packages, DESeq wants its input in a very particular format

```
dds <- DESeqDataSetFromMatrix(countData = counts,
                       colData = metadata,
                       design = ~dex)
```

```
converting counts to integer mode

Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
design formula are characters, converting to factors
```

```
dds <- DESeq(dds)
```

```
estimating size factors

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

fitting model and testing
```

```
res <- results(dds)
head(res)
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 6 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
```

```
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
                      padj
                 <numeric>
ENSG00000000003  0.163035
ENSG00000000005        NA
ENSG00000000419  0.176032
ENSG00000000457  0.961694
ENSG00000000460  0.815849
ENSG00000000938        NA
```

Save results

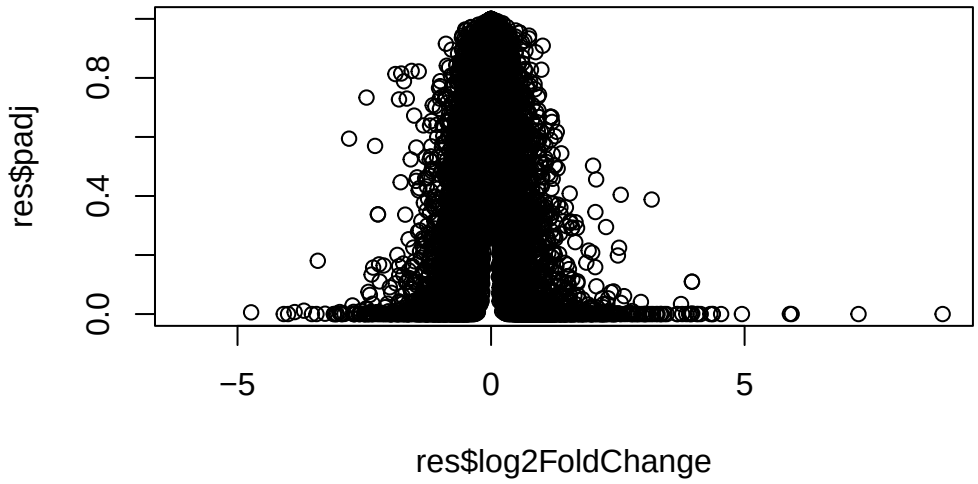
Save out the results to CSV file

```
write.csv(res, file = "myresults.csv")
```

Volcano plots

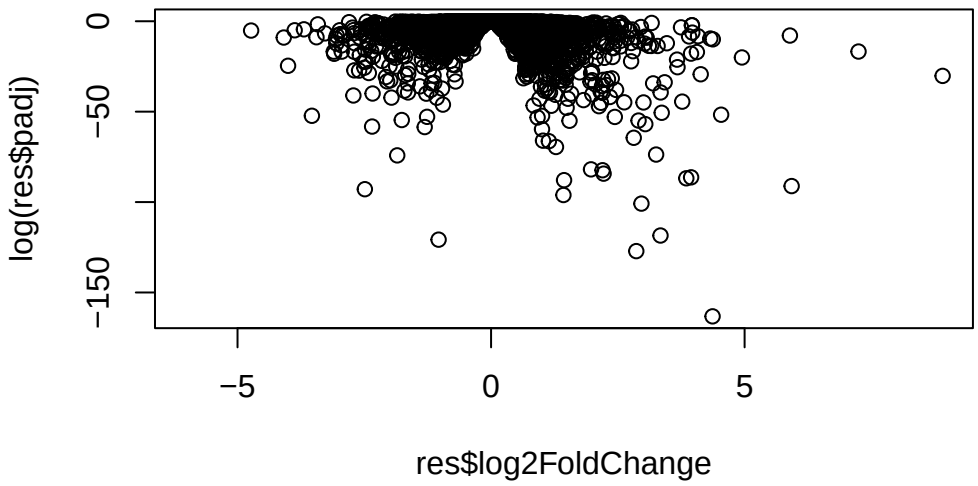
Let's make a common summary plot of our results Our main results here are the log2 fold change and adjusted p value

```
plot(res$log2FoldChange, res$padj)
```



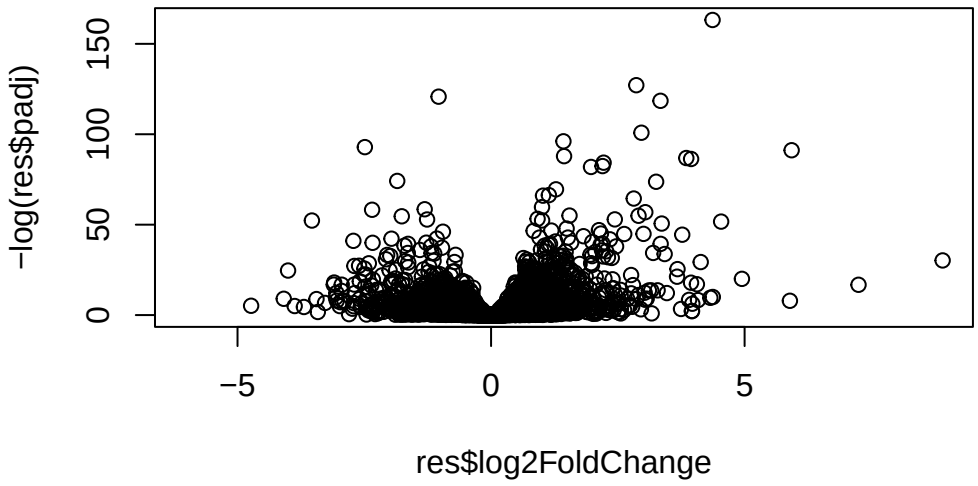
We need to transform the p value axis here so we can see the data we actually care about (small p value)

```
plot(res$log2FoldChange, log(res$padj))
```



To make the plot better, we need to flip the plot on the y axis so the most significant values are towards the top

```
plot(res$log2FoldChange, -log(res$padj))
```

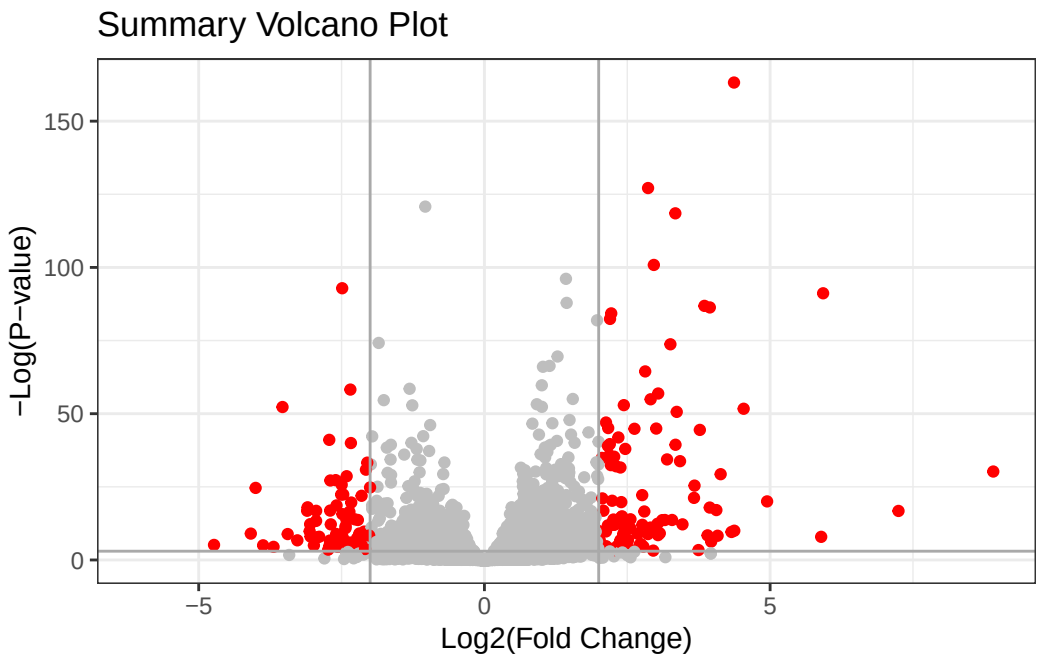


This is our standard volcano plot, but let's make it more interesting in ggplot. We can use color to highlight the most important subset of transcripts with a log2FC > +2 and < -2 that have a P value < 0.05. We will need a custom color vector for this

```
mycols <- rep("gray", nrow(res))
mycols[res$log2FoldChange >= 2] <- "red"
mycols[res$log2FoldChange <= -2] <- "red"
mycols[res$padj >= 0.05] <- "gray"
```

```
ggplot(res) +
  aes(log2FoldChange, -log(padj)) +
  geom_point(col=mycols) +
  labs(title = "Summary Volcano Plot", x = "Log2(Fold Change)", y = "-Log(P-value)") +
  theme_bw() +
  geom_vline(xintercept = c(-2,2), col = "darkgray") +
  geom_hline(yintercept = -log(0.05), col = "darkgray")
```

```
Warning: Removed 23549 rows containing missing values or values outside the scale range
(`geom_point()`).
```



Adding annotation data

At the minute all we know about the genes in our dataset is their ENSEMBLE database id

```
head(rownames(res))
```

```
[1] "ENSG00000000003" "ENSG00000000005" "ENSG00000000419" "ENSG00000000457"
[5] "ENSG00000000460" "ENSG00000000938"
```

We can use a set of BioConductor packages to amp these ENSEMBLE ids to things like GENE SYMBOL, REFSEQ id, ENTREZ id, etc. In other words, what each gene is called in different databases that I might use to work with for downstream packages.

I install these packages with BiocManager::install()

```
library("AnnotationDbi")
library("org.Hs.eg.db")
```

the different formats that I can covert IDs between include:

```
columns(org.Hs.eg.db)
```

```
 [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
 [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"  "GENENAME"
[11] "GENETYPE"     "GO"           "GOALL"        "IPI"          "MAP"
[16] "OMIM"         "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"
[21] "PMID"         "PROSITE"      "REFSEQ"       "SYMBOL"       "UCSCKG"
[26] "UNIPROT"
```

We can use the mapIds() function to do this

```
res$symbol <- mapIds(org.Hs.eg.db,
                     keys=row.names(res), # Our genenames
                     keytype="ENSEMBL", # The format of our genenames
                     column="SYMBOL", # The new format we want to add
                     multiVals="first")
```

```
'select()' returned 1:many mapping between keys and columns
```

```
head(res)
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 7 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
                     padj      symbol
                <numeric> <character>
ENSG00000000003  0.163035      TSPAN6
ENSG00000000005        NA        TNMD
ENSG00000000419  0.176032        DPM1
ENSG00000000457  0.961694       SCYL3
ENSG00000000460  0.815849       FIRRM
ENSG00000000938        NA         FGR
```

Q11. Run the mapIds() function two more times to add the Entrez ID and UniProt accession and GENENAME as new columns called res$entrez, res$uniprot and res$genename.

```
res$entrez <- mapIds(org.Hs.eg.db,
                     keys=row.names(res), # Our genenames
                     keytype="ENSEMBL", # The format of our genenames
                     column="ENTREZID", # The new format we want to add
                     multiVals="first")
```

```
'select()' returned 1:many mapping between keys and columns
```

```
res$genename <- mapIds(org.Hs.eg.db,
                     keys=row.names(res), # Our genenames
                     keytype="ENSEMBL", # The format of our genenames
                     column="GENENAME", # The new format we want to add
                     multiVals="first")
```

```
'select()' returned 1:many mapping between keys and columns
```

```
res$uniprot <- mapIds(org.Hs.eg.db,
                      keys=row.names(res), # Our genenames
                      keytype="ENSEMBL", # The format of our genenames
                      column="UNIPROT", # The new format we want to add
                      multiVals="first")
```

```
'select()' returned 1:many mapping between keys and columns
```

```
head(res)
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 10 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
                     padj      symbol      entrez              genename
                <numeric> <character> <character>           <character>
ENSG00000000003  0.163035      TSPAN6        7105          tetraspanin 6
ENSG00000000005        NA        TNMD       64102            tenomodulin
ENSG00000000419  0.176032        DPM1        8813 dolichyl-phosphate m..
ENSG00000000457  0.961694       SCYL3       57147 SCY1 like pseudokina..
ENSG00000000460  0.815849       FIRRM       55732 FIGNL1 interacting r..
ENSG00000000938        NA         FGR        2268 FGR proto-oncogene, ..
                   uniprot
                <character>
ENSG00000000003  A0A087WYV6
ENSG00000000005      Q9H2S6
ENSG00000000419      H0Y368
ENSG00000000457      X6RHX1
ENSG00000000460      A6NFP1
ENSG00000000938      B7Z6W7
```

```
write.csv(res, file = "myresults_annotated.csv")
```

Pathway Analysis

Let's use KEGG to see which pathway my gene sets overlap with (ie highlight the biology that may be influenced by the dex drug treatment)

We will use the following packages: BiocManager::install(c("pathview", "gage", "gageData"))

The gage function wants as input a "named vector of importance"

```
foldchanges = res$log2FoldChange
names(foldchanges) = res$entrez
```

```
library(pathview)
```

```
################################################################################
Pathview is an open source software package distributed under GNU General
Public License version 3 (GPLv3). Details of GPLv3 is available at
http://www.gnu.org/licenses/gpl-3.0.html. Particullary, users are required to
formally cite the original Pathview paper (not just mention it) in publications
or products. For details, do citation("pathview") within R.

The pathview downloads and uses KEGG data. Non-academic uses may require a KEGG
license agreement (details at http://www.kegg.jp/kegg/legal.html).
################################################################################
```

```
library(gage)
```

```
library(gageData)

data(kegg.sets.hs)
keggres = gage(foldchanges, gsets=kegg.sets.hs)
```

```
head(keggres$less)
```

```
                                              p.geomean stat.mean
hsa05332 Graft-versus-host disease          0.0004250461 -3.473346
hsa04940 Type I diabetes mellitus           0.0017820293 -3.002352
```

```
hsa05310 Asthma                                          0.0020045888 -3.009050
hsa04672 Intestinal immune network for IgA production 0.0060434515 -2.560547
hsa05330 Allograft rejection                             0.0073678825 -2.501419
hsa04340 Hedgehog signaling pathway                      0.0133239547 -2.248547
                                                                p.val      q.val
hsa05332 Graft-versus-host disease                       0.0004250461 0.09053483
hsa04940 Type I diabetes mellitus                        0.0017820293 0.14232581
hsa05310 Asthma                                          0.0020045888 0.14232581
hsa04672 Intestinal immune network for IgA production 0.0060434515 0.31387180
hsa05330 Allograft rejection                             0.0073678825 0.31387180
hsa04340 Hedgehog signaling pathway                      0.0133239547 0.47300039
                                                         set.size         exp1
hsa05332 Graft-versus-host disease                             40 0.0004250461
hsa04940 Type I diabetes mellitus                              42 0.0017820293
hsa05310 Asthma                                                29 0.0020045888
hsa04672 Intestinal immune network for IgA production          47 0.0060434515
hsa05330 Allograft rejection                                   36 0.0073678825
hsa04340 Hedgehog signaling pathway                            56 0.0133239547
```

We can have a quick look at one of the highlighted pathways

```
pathview(gene.data=foldchanges, pathway.id="hsa05310")
```

```
'select()' returned 1:1 mapping between keys and columns
```

```
Info: Working in directory C:/Users/iruud/BGGN213/class_13
```

```
Info: Writing image file hsa05310.pathview.png
```